

```
1 import pandas as pd
2 from pandas import DataFrame
3 from math import log
4 from collections import Counter
5 from pprint import pprint
6
7 df_tennis = pd.read_csv('/home/sahyadri/Desktop/rantest.csv')
8 df_tennis.keys()[0]
9
10 def entropy(probs):
11     return sum([-prob * log(prob,2)for prob in probs])
12
13 def entropy_of_list(a_list):
14     cnt=Counter(x for x in a_list)
15     num_instances= len(a_list)*1.0
16     probs=[x/num_instances for x in cnt.values()]
17     return entropy(probs)
18 def information_gain(df,split_attribute_name,target_attribute_name):
19     df_split = df.groupby(split_attribute_name)
20     nobs= len(df.index)*1.0
21     df_agg_cnt=df_split.agg({target_attribute_name:[entropy_of_list,lambda x:len(x)/nobs]})[target_attribute_name]
22     df_agg_cnt.columns = ['Entropy','PropObservations']
23     new_entropy=sum(df_agg_cnt['Entropy']*df_agg_cnt['PropObservations'])
24     old_entropy=entropy_of_list(df[target_attribute_name])
25     return old_entropy - new_entropy
26
27 def id3(df,target_attribute_name,attribute_names,default_class=None):
28     cnt=Counter(x for x in df[target_attribute_name])
29     if len(cnt)==1:
30         return next(iter(cnt))
31     elif df.empty or(not attribute_names):
32         return default_class
33     else:
34         default_class = max(cnt.keys())
35         best_attribute_name = None
36         best_info_gain = -1
37         for attribute_name in attribute_names:
38             info_gain = information_gain(df,attribute_name,target_attribute_name)
39             if info_gain > best_info_gain:
40                 best_info_gain = info_gain
41                 best_attribute_name = attribute_name
42         attribute_names.remove(best_attribute_name)
43         tree = {best_attribute_name:{}}
44         for value in df[best_attribute_name].unique():
45             subtree = id3(df[df[best_attribute_name]==value],target_attribute_name,attribute_names,default_class)
46             tree[best_attribute_name][value]=subtree
47         return tree
```

```

30     return next(iter(cnt))
31 elif df.empty or(not attribute_names):
32     return default_class
33 else:
34     default_class = max(cnt.keys())
35     gainz=[information_gain(df,attr,target_attribute_name) for attr in attribute_names]
36     index_of_max=gainz.index(max(gainz))
37     best_attr=attribute_names[index_of_max]
38     tree={best_attr:{}}
39     remaining_attribute_names=[i for i in attribute_names if i!=best_attr]
40
41     for attr_val,data_subset in df.groupby(best_attr):
42         subtree=id3(data_subset,target_attribute_name,remaining_attribute_names,default_class)
43         tree[best_attr][attr_val]=subtree
44     return tree
45 attribute_names=list(df_tennis.columns)
46 attribute_names.remove('PlayTennis')
47 tree=id3(df_tennis,'PlayTennis',attribute_names)
48 print("\n\nThe resultant Decision Tree is:\n")
49 pprint(tree)
50

```

The resultant Decision Tree is:

```

{'OutLook': {'Overcast': 'Yes',
    'Rain': {'Wind': {'Strong': 'No', 'Weak': 'Yes', 'Weak ': 'Yes'}},
    'Sunny': {'Humidity': {'High': 'No', 'Normal': 'Yes'}}}}

```

In [2]:

```

1 import pandas as pd
2 from pandas import DataFrame
3 from math import log
4 from collections import Counter
5 from pprint import pprint
6
7 df_tennis = pd.read_csv('/home/sahyadri/Desktop/rantest.csv')
8 df_tennis.keys()[0]
9
10 def entropy(probs):
11     return sum([-prob * log(prob,2)for prob in probs])
12
13 def entropy_of_list(a_list):
14     cnt=Counter(x for x in a_list)
15     num_instances= len(a_list)*1.0
16     probs=[x/num_instances for x in cnt.values()]
17     return entropy(probs)
18 def information_gain(df,split_attribute_name,target_attribute_name):
19     df_split = df.groupby(split_attribute_name)
20     nobs= len(df.index)*1.0
21     df_agg_cnt=df_split.agg({target_attribute_name:[entropy_of_list,lambda x:len(x)/nobs]} )[target_attribute_name]
22     df_agg_cnt.columns = ['Entropy','PropObservations']
23     new_entropy=sum(df_agg_cnt['Entropy']*df_agg_cnt['PropObservations'])
24     old_entropy=entropy_of_list(df[target_attribute_name])
25     return old_entropy - new_entropy
26
27 def id3(df,target_attribute_name,attribute_names,default_class=None):
28     cnt=Counter(x for x in df[target_attribute_name])
29     if len(cnt)==1:
30         return next(iter(cnt))
31     elif df.empty or(not attribute_names):
32         return default_class
33     else:
34         default_class = max(cnt.keys())
35         gainz=[information_gain(df,attr,target_attribute_name) for attr in attribute_names]
36         index_of_max=gainz.index(max(gainz))
37         best_attr=attribute_names[index_of_max]
38         tree={best_attr:{}}
39         remaining_attribute_names=[i for i in attribute_names if i!=best_attr]
40
41         for attr_val,data_subset in df.groupby(best_attr):
42             subtree=id3(data_subset,target_attribute_name,remaining_attribute_names,default_class)
43             tree[best_attr][attr_val]=subtree
44     return tree
45 attribute_names=list(df_tennis.columns)
46 attribute_names.remove('PlayTennis')
47 tree=id3(df_tennis,'PlayTennis',attribute_names)
48 print("The resultant Decision Tree is:\n")
49 pprint(tree)
50

```

The resultant Decision Tree is:

```
{'OutLook': {'Overcast': 'Yes':
```

```

In [2]: 1 import pandas as pd
2 from pandas import DataFrame
3 from math import log
4 from collections import Counter
5 from pprint import pprint
6
7 df_tennis = pd.read_csv('/home/sahyadri/Desktop/rantest.csv')
8 df_tennis.keys()[0]
9
10 def entropy(probs):
11     return sum([-prob * log(prob,2) for prob in probs])
12
13 def entropy_of_list(a_list):
14     cnt=Counter(x for x in a_list)
15     num_instances= len(a_list)*1.0
16     probs=[x/num_instances for x in cnt.values()]
17     return entropy(probs)
18
19 def information_gain(df,split_attribute_name,target_attribute_name):
20     df_split = df.groupby(split_attribute_name)
21     nobs=[len(df.index)*1.0]
22     df_agg_cnt=df_split.agg({target_attribute_name:[entropy_of_list,lambda x:len(x)/nobs]})(target_attribute_name)
23     df_agg_cnt.columns = [ 'Entropy', 'PropObservations']
24     new_entropy=sum(df_agg_cnt['Entropy']/df_agg_cnt['PropObservations'])
25     old_entropy=entropy_of_list(df[target_attribute_name])
26     return old_entropy - new_entropy
27
28 def id3(df,target_attribute_name,attribute_names,default_class=None):
29     cnt=Counter(x for x in df[target_attribute_name])
30     if len(cnt)==1:
31         return next(iter(cnt))
32     elif df.empty or (not attribute_names):
33         return default_class
34     else:
35         default_class = max(cnt.keys())
36         gainz=[information_gain(df,attr,target_attribute_name) for attr in attribute_names]
37         index_of_max=gainz.index(max(gainz))
38         best_attr=attribute_names[index_of_max]
39         tree={best_attr:{}}
40         remaining_attribute_names=[i for i in attribute_names if i!=best_attr]
41
42         for attr_val,data_subset in df.groupby(best_attr):
43             subtree=id3(data_subset,target_attribute_name,remaining_attribute_names,default_class)
44             tree[best_attr][attr_val]=subtree
45
46     return tree
47 attribute_names=list(df_tennis.columns)
48 attribute_names.remove('PlayTennis')
49 tree=id3(df_tennis,'PlayTennis',attribute_names)
50 print("\n\nThe resultant Decision Tree is:\n")
51 pprint(tree)

```

The resultant Decision Tree is:

```
{"Outlook": {"Overcast": "Yes",
    "Rain": {"Wind": {"Strong": "No", "Weak": "Yes", "Weak": "Yes"}},
    "Sunny": {"Humidity": {"High": "No", "Normal": "Yes"}}})
```

In [2]:

```
1 import pandas as pd
2 from pandas import DataFrame
3 from math import log
4 from collections import Counter
5 from pprint import pprint
6
7 df_tennis = pd.read_csv('/home/sahyadri/Desktop/rantest.csv')
8 df_tennis.keys()[0]
9
10 def entropy(probs):
11     return sum([-prob * log(prob,2)for prob in probs])
12
13 def entropy_of_list(a_list):
14     cnt=Counter(x for x in a_list)
15     num_instances= len(a_list)*1.0
16     probs=[x/num_instances for x in cnt.values()]
17     return entropy(probs)
18
19 def information_gain(df,split_attribute_name,target_attribute_name):
20     df_split = df.groupby(split_attribute_name)
21     nobs= len(df.index)*1.0
22     df_agg_cnt=df_split.agg({target_attribute_name:[entropy_of_list,lambda x:len(x)/nobs]})[target_attribute_name]
23     df_agg_cnt.columns = ['Entropy','PropObservations']
24     new_entropy=sum(df_agg_cnt['Entropy']*df_agg_cnt['PropObservations'])
25     old_entropy=entropy_of_list(df[target_attribute_name])
26
27     return old_entropy - new_entropy
28
29 def id3(df,target_attribute_name,attribute_names,default_class=None):
30     cnt=Counter(x for x in df[target_attribute_name])
31     if len(cnt)==1:
32         return next(iter(cnt))
33     elif df.empty or(not attribute_names):
34         return default_class
35     else:
36         default_class = max(cnt.keys())
37
38         best_attribute = select_attribute(attribute_names,df,target_attribute_name)
39
40         tree = {best_attribute:{}}
41
42         for value in df[best_attribute].unique():
43             subtree = id3(df[df[best_attribute]==value],target_attribute_name,attribute_names,value)
44             tree[best_attribute][value]=subtree
45
46         return tree
```

this network before you can access the Internet. [Open network login page](#)



Logout

jupyter exp4 Last Checkpoint: Last Friday at 10:16 AM (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted

Python 3 C

```
33     else:
34         default_class = max(cnt.keys())
35         gainz=[information_gain(df,attr,target_attribute_name) for attr in attribute_names]
36         index_of_max=gainz.index(max(gainz))
37         best_attr=attribute_names[index_of_max]
38         tree={best_attr:{}}
39         remaining_attribute_names=[i for i in attribute_names if i!=best_attr]
40
41         for attr_val,data_subset in df.groupby(best_attr):
42             subtree=id3(data_subset,target_attribute_name,remaining_attribute_names,default_class)
43             tree[best_attr][attr_val]=subtree
44
45     attribute_names=list(df_tennis.columns)
46     attribute_names.remove('PlayTennis')
47     tree=id3(df_tennis,'PlayTennis',attribute_names)
48     print("\n\nThe resultant Decision Tree is:\n")
49     pprint(tree)
50
```

The resultant Decision Tree is:

```
{'OutLook': {'Overcast': 'Yes',
              'Rain': {'Wind': {'Strong': 'No', 'Weak': 'Yes', 'Weak ': 'Yes'}},
              'Sunny': {'Humidity': {'High': 'No', 'Normal': 'Yes'}}}}
```

In []:

1

ⓘ You must log in to this network before you can access the Internet.

[Open network login page](#)

jupyter exp4



Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted



Python 3

File Edit View Insert Cell Kernel Widgets Help

```
28 in df[target_attribute_name])
29
30 (cnt))
31 attribute_names):
32 lass
33
34 max(cnt.keys())
35 on_gain(df,attr,target_attribute_name) for attr in attribute_names]
36 nz.index(max(gainz))
37 ute_names[index_of_max]
38 {}
39 ute_names=[i for i in attribute_names if i!=best_attr]
40
41 a_subset in df.groupby(best_attr):
42 (data_subset,target_attribute_name,remaining_attribute_names,default_class)
43 tr][attr_val]=subtree
44
45 tennis.columns)
46 PlayTennis')
47 Tennis',attributes_name)
48 Decision Tree is:\n")
49
50
```